Milestone Report

Andy Tse

Overview

- For this report, we are going to perform an analysis on using the data that is provided by Swiftkey.
- It consists of three different files from different media outlets: news, blogs, and Twitter.
- The goal is to utilize the exploratory data analysis to determine the next word that is predicted.
- It comes in four different languages: English, Finnish, Russian, and German. In this project, the English version is going to be used in the analysis.

Getting Working Directory

```
getwd()
setwd("./Data Science Capstone")
```

Loading All the Libraries

In this component, we are going to load all the necessary packages that are going to be used in the study.

```
library(knitr)
library(tm)
library(NLP)
library(plyr)
library(SnowballC)
library(RWeka)
library(wordcloud)
library(slam)
library(stringi)
library(ggplot2)
library(dplyr)
library(R.utils)
library(openNLP)
library(textmining)
```

Loading Raw Datasets and Reading the Data

For this part, we are going to read the datasets from all media outlets. This is based off the text files that is provided from Swiftkey.

```
conn <- file("en_US.blogs.txt", open = "rb")
blogs <- readLines(conn, encoding = "UTF-8")
close(conn)

# Read news data in binary mode
conn <- file("en_US.news.txt", open = "rb")
news <- readLines(conn, encoding = "UTF-8")
close(conn)

# Read twitter data in binary mode
conn <- file("en_US.twitter.txt", open = "rb")
twits <- readLines(conn, encoding = "UTF-8")
close(conn)

rm(conn)
```

Analyzing Datasets

For this part, we are going to use the code in order to retrieve the sample data for the amount of words, lines, and other statistical component to get the information. We are gathering the data from all media outlets.

```
# Compute words per line info on each line for each data type
rawWPL<-lapply(list(blogs,news,twits),function(x) stri_count_words(x))

# Compute statistics and summary info for each data type
rawstats<-data.frame(
  File=c("blogs","news","twitter"),
  t(rbind(sapply(list(blogs,news,twits),stri_stats_general),
          TotalWords=sapply(list(blogs,news,twits),stri_stats_latex)
[4,])),
  # Compute words per line summary

WPL=rbind(summary(rawWPL[[1]]),summary(rawWPL[[2]]),summary(rawWPL[[3]]
))
)
print(rawstats)
```

Below is the summary of all the lines for all media outlets on the data that with total lines, characters, empty lines, and words used in the files. In this information set, blogs contain 899,288 lines, the news has 1,010,242 lines, and Twitter contains 2,360,148 lines. It has been estimated that the blogs have the most for the total words, while Twitter has the most lines.

```
     File   Lines LinesNEmpty     Chars CharsNWhite TotalWords WPL.Min.
WPL.1st.Qu. WPL.Median WPL.Mean
1   blogs  899288      899288 206824382   170389539   37570839        0
9          28    41.75
2    news 1010242     1010242 203223154   169860866   34494539        1
19          32    34.41
3 twitter 2360148     2360148 162096031   134082634   30451128        1
7          12    12.75
  WPL.3rd.Qu. WPL.Max.
1          60     6726
2          46     1796
3          18       47
```

Sample Data

We will now take a sample size of 20000 and create a random generated seed in this dataset. It will be run on a random 5% sample.

```
samplesize <- 20000
set.seed(12345)
twitSample <- twits[rbinom(length(twits)*0.05,length(twits),0.5)]
twitSample <- iconv(twitSample,'UTF-8', 'ASCII', "byte")
newsSample <- news[rbinom(length(news)*0.05,length(news),0.5)]
newsSample <- iconv(newsSample,'UTF-8', 'ASCII', "byte")
blogsSample <- blogs[rbinom(length(blogs)*0.05,length(news),0.5)]
blogsSample <- iconv(blogsSample,'UTF-8', 'ASCII', "byte")

text.Sample <- paste(twitSample,newsSample,blogsSample)

text.Sample <- Corpus(VectorSource(text.Sample))
```

Cleaning the Data

For this section, we are going clean and tidy up the data that is not useful for the analytical study. We are going to remove the unnecessary characteristics and non-English words as well.

```

# Lover Case Letter Conversion
text.Sample<- tm_map(text.Sample, tolower)

# Punctuation Removal in Text
text.Sample<- tm_map(text.Sample, removePunctuation)

# Numbers Removal in Text
text.Sample<- tm_map(text.Sample, removeNumbers)

## White Space Removal
```

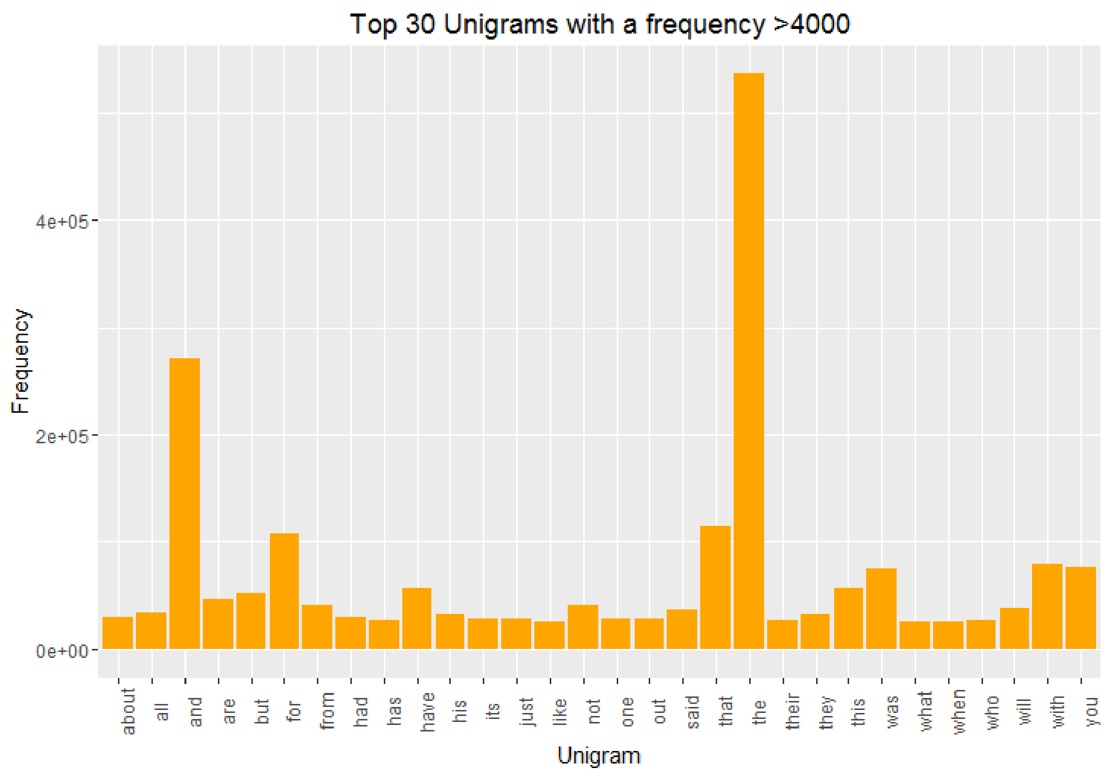
```
text.Sample <- tm_map(text.Sample, stripWhitespace)

## Plaintext Document Removal
text.Sample <- tm_map(text.Sample, PlainTextDocument)
```

Histograms

In this section below, we are going to create the histogram charts that predicts all the words in different categories. We are creating charts for the Unigram, Bigram, and Trigram analyses for frequently used words. Based on the information off the charts, we are only going to focus on the top 30 words that are used in the analysis based on what people have said in the media.

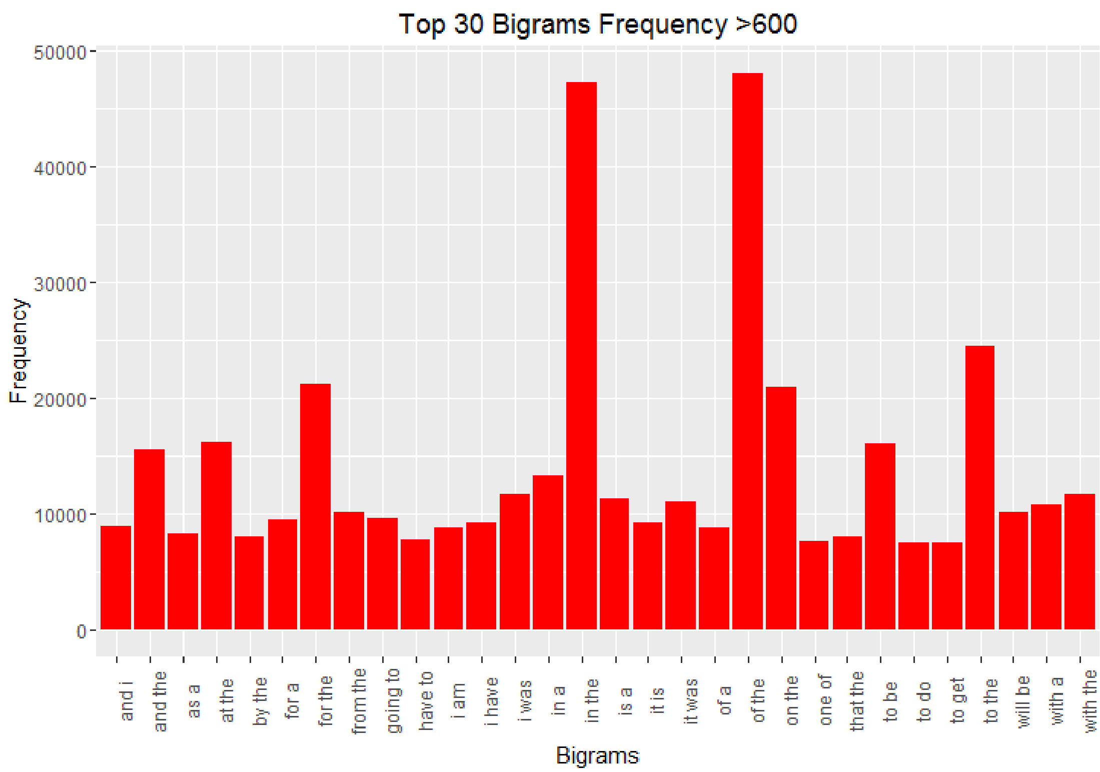
Unigrams

```
UnigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1,
max = 1))

text.Sample.Unigram <- TermDocumentMatrix(text.Sample, control =
list(tokenize = UnigramTokenizer))

FreqTerms <- findFreqTerms(text.Sample.Unigram, lowfreq = 4000)

text.Sample.Frequency.Vector.Uni <-
sort(rowSums(as.matrix(text.Sample.Unigram[FreqTerms,])),decreasing=TRU
E)
text.Sample.Frequency.Dataframe.Uni <- data.frame(word =
names(text.Sample.Frequency.Vector.Uni),freq=text.Sample.Frequency.Vect
or.Uni)

ggplot(data=text.Sample.Frequency.Dataframe.Uni[1:30,],aes(x=word,y=fre
q)) +
  geom_bar(stat="identity", fill="orange")  +
theme(axis.text.x=element_text(angle=90)) +
  labs(title="Top 30 Unigrams with a frequency >4000") +
  labs(x="Unigram") + labs(y="Frequency")
```



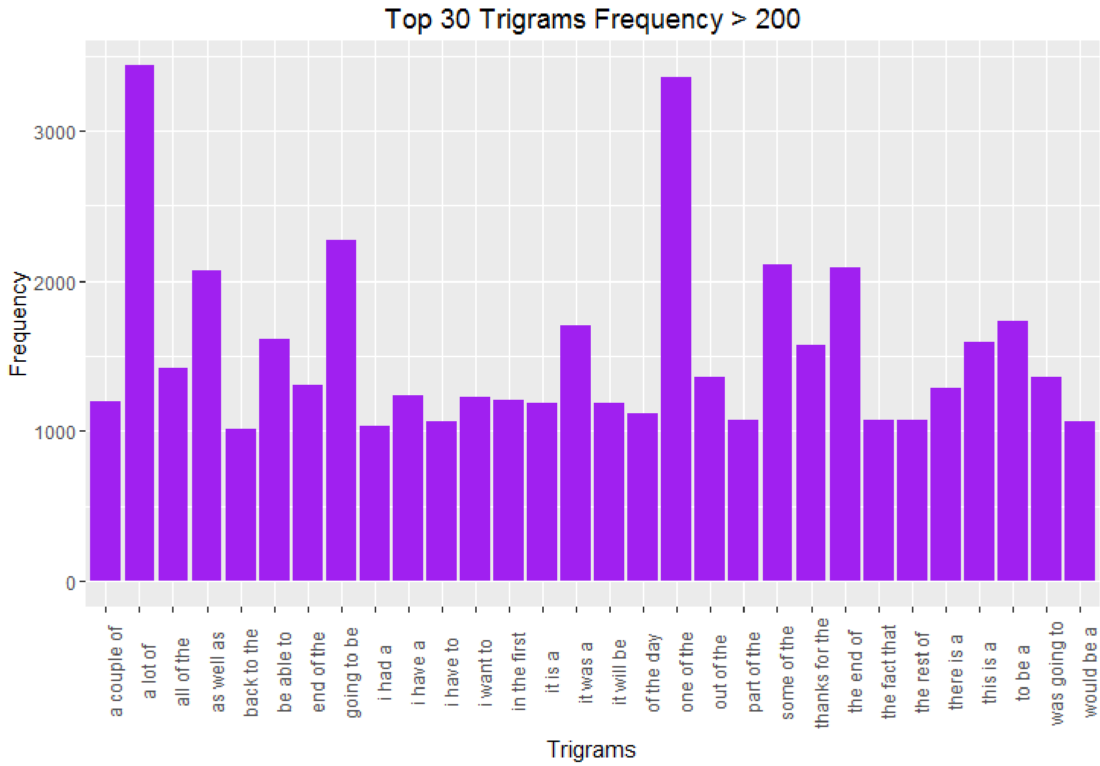
Bigrams

```
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2,
max = 2))

text.Sample.Bigram <- TermDocumentMatrix(text.Sample, control =
list(tokenize = BigramTokenizer))

FreqTerms <- findFreqTerms(text.Sample.Bigram, lowfreq = 600)

text.Sample.Frequency.Vector.Bi <-
sort(rowSums(as.matrix(text.Sample.Bigram[FreqTerms,])),decreasing=TRUE
)

text.Sample.Frequency.Dataframe.Bi <- data.frame(word =
names(text.Sample.Frequency.Vector.Bi),freq=text.Sample.Frequency.Vecto
r.Bi)

ggplot(data=text.Sample.Frequency.Dataframe.Bi[1:30,],aes(x=word,y=freq
)) +
  geom_bar(stat="identity", fill="red") +
theme(axis.text.x=element_text(angle=90)) +
  labs(title="Top 30 Bigrams Frequency >600") +
  labs(x="Bigrams") + labs(y="Frequency")
```



Trigrams

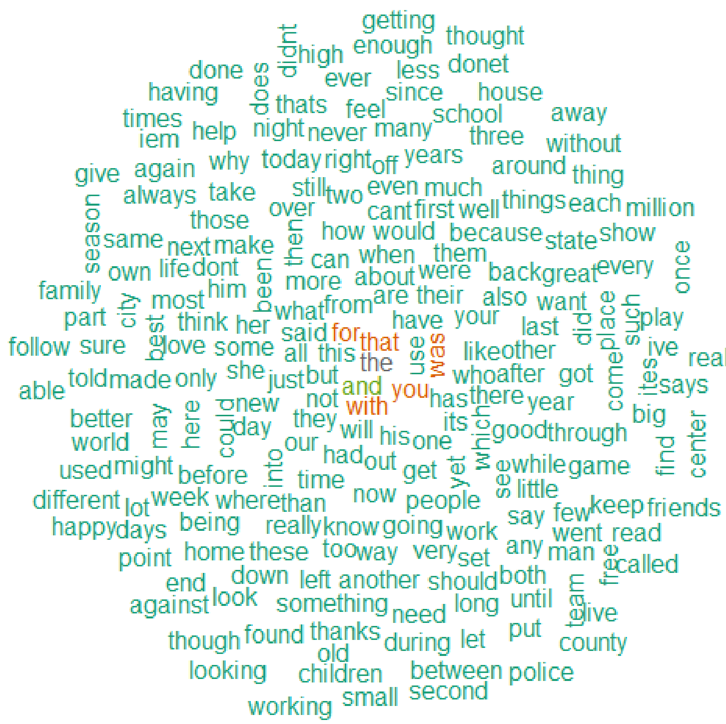
```
TrigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 3,
max = 3))

text.Sample.Trigram <- TermDocumentMatrix(text.Sample, control =
list(tokenize = TrigramTokenizer))

FreqTerms <- findFreqTerms(text.Sample.Trigram, lowfreq = 200)

text.Sample.Frequency.Vector.Tri <-
sort(rowSums(as.matrix(text.Sample.Trigram[FreqTerms,])),decreasing=TRU
E)
text.Sample.Frequency.Dataframe.Tri <- data.frame(word =
names(text.Sample.Frequency.Vector.Tri),freq=text.Sample.Frequency.Vect
or.Tri)

ggplot(data=text.Sample.Frequency.Dataframe.Tri[1:30,],aes(x=word,y=fre
q)) +
  geom_bar(stat="identity", fill="purple") +
theme(axis.text.x=element_text(angle=90)) +
  labs(title="Top 30 Trigrams Frequency > 200") +
  labs(x="Trigrams") + labs(y="Frequency")
```



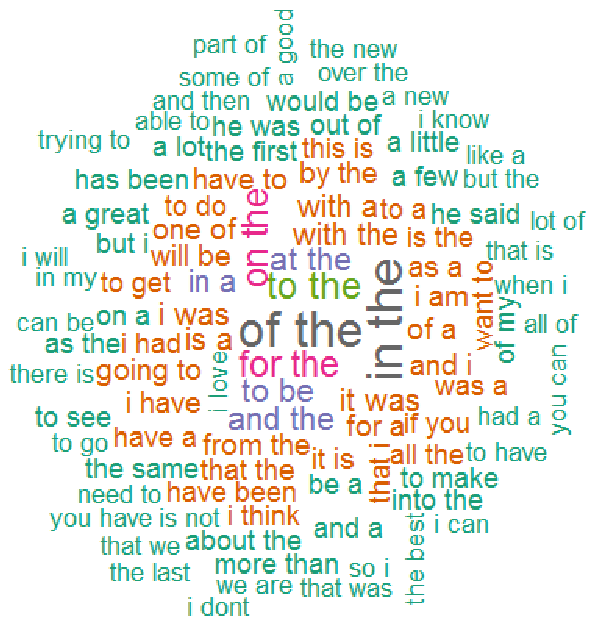
Wordcloud

In this section, the analysis will be drawn on which string of words that people are going to say in different media outlets for this model. It is going to be divided in different n-gram model categories.

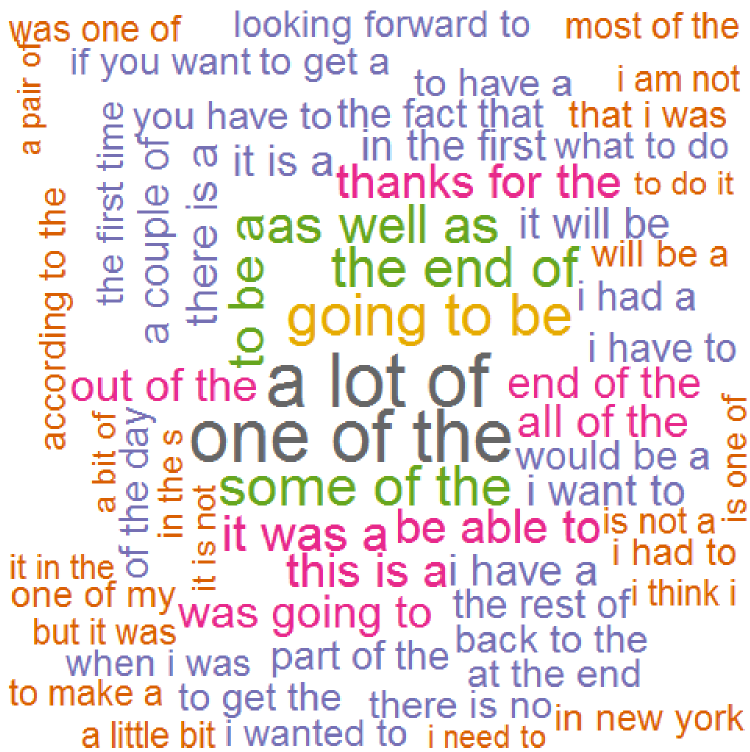
```
wordcloud(words=text.Sample.Frequency.Dataframe.Uni$word,max.words =
300, freq= text.Sample.Frequency.Dataframe.Uni$freq, scale = c(1,1),
random.order = F, colors =brewer.pal(20, "Dark2"))
```



```
wordcloud(words=text.Sample.Frequency.Dataframe.Bi$word,max.words =
100, freq= text.Sample.Frequency.Dataframe.Bi$freq, scale = c(2,1),
random.order = F, colors =brewer.pal(10, "Dark2"))
```




```
wordcloud(words=text.Sample.Frequency.Dataframe.Tri$word,max.words =
200, freq= text.Sample.Frequency.Dataframe.Tri$freq, scale = c(3,1),
random.order = F, colors =brewer.pal(10, "Dark2"))
```



Next Steps for the Project

- With this analysis done, the next step is to create a shiny app to make the next word prediction based on the information that is provided in the report.
- The dataframes would be used to calculate its probability on the words that are used next.